Towards Comprehensive and Collaborative Forensics on Email Evidence[†]

Justin Paglierani, Mike Mabey and Gail-Joon Ahn
Arizona State University
{jpaglier,mmabey,gahn}@asu.edu

***Abstract*—The digital forensics community has neglected email forensics *as a process*, despite the fact that email remains an important tool in the commission of crime. At present, there exists little support for discovering, acquiring, and analyzing web-based email, despite its widespread use. In this paper we present a systematic process for email forensics which we integrate into the normal forensic analysis workflow, and which accommodates the distinct characteristics of email evidence. Our process focuses on detecting the presence of non-obvious artifacts related to email accounts, retrieving the data from the service provider, and representing email in a well-structured format based on existing standards. As a result, developers and organizations can collaboratively create and use analysis tools that can analyze email evidence from any source in the same fashion and the examiner can access additional data relevant to their forensic cases.**

***Index Terms*—Email, forensics, collaboration.**

I. Introduction

The recent investigation of a senior U.S. intelligence official reaffirmed the importance of email forensics [1]. The investigation relied on the simple inspection of the drafts folder of a shared email account, but if the suspects had taken the time to make their correspondence more clandestine, a more sophisticated approach would have been necessary to discover relevant email evidence. Current methodologies do not address the possible intricacies introduced when an investigation centers around the analysis of various email sources and, furthermore, simple inspection does not aid an examiner in detecting the presence of email which is not locally stored [2, pg. 471].

Consider a scenario in which a suspected computer criminal has communicated with many parties about the nature and means of their actions using various communication methods, including locally stored emails and webmail accounts. When an examiner seizes a suspect's hard drive, only the locally stored or cached email would be directly available and the webmail accounts would remain undiscovered without substantial manual effort. These missing portions of data could lead to an incomplete investigation report with respect to the suspected act. Even if evidence resides locally in diverse formats, it is likely that an examiner would need separate tools and methods to analyze each of them.

In addition, an emerging notion called the *Internet of Things* [3] (IoT) presents an environment in which any number of "things," or devices, connect to each other through the Internet. Such a model clearly presents an even greater challenge for attempting manual analysis of device data or behavior as part of an investigation.

We aim to provide a systematic and forensically sound methodology for discovering, extracting, and analyzing evidence from email which resides through emerging technologies. Our approach has several unique properties to support collaborative email forensics. First, we propose a process-driven email forensics approach comprised of several sub-tasks for collaboratively analyzing email evidence. Since each examiner has different capabilities, this process-driven workflow can help them conduct forensics tasks more efficiently and effectively by reducing the number of backlogged cases and allowing for the sharing of work in a collaborative manner. Second, we attempt to build pluggable modules so that our framework can be realized as a service to multiple examiners without asking them to alter their forensic environment or tools. Third, multiple examiners and tools have their own regulatory report and proprietary data format that have been critical barriers to them collaborating with each other. Hence, we introduce an extended XML-based format to represent email evidence with a uniform and interoperable evidence container for collaborative email forensics. We hope that with the aid of our approach, the digital forensics community can begin establishing best practice standards to acquire, process, authenticate, analyze, and present this distinct family of evidence.

A. A Note on Legality

We emphasize that our approach requires special consideration to laws regarding the search and seizure of evidence. In many territories, it may be necessary to secure a subpoena or warrant before using an approach like ours. However, in the event that the necessary procedures have been followed and the service provider remains uncooperative, our method provides examiners with an alternative means of acquisition for the sake of prompt response, as discussed by Richard Littlehale[1] in his testimony before the U.S. House Judiciary Subcommittee on Crime, Terrorism, Homeland Security &

[†]This work was partially supported by the grant from National Science Foundation. All correspondences should be addressed to: Dr. Gail-Joon Ahn at gahn@asu.edu.

[1] Assistant Special Agent in Charge, Technical Services Unit, Tennessee Bureau of Investigation.

TABLE I: Predicted daily email traffic in billions from 2012-2016 as published by the Radicati Group

Year	2012	2013	2014	2015	2016
Total worldwide emails/day (B)	144.8	154.6	165.8	178.3	192.2
% Change	—	7%	7%	8%	8%

Investigations on March 19, 2013 [4]. We urge practitioners to consult proficient legal counsel before utilizing the information contained herein.

II. Related Work

In case there was ever any doubt as to how important email is to private and corporate communication, the Executive Summary [5] of the Radicati Group's report titled "Email Market, 2012-2016" states that the total number of worldwide emails sent each day in 2012 was about 144.8 billion, with steady growth predicted for years to come as shown in Table I. Furthermore, the report states that "the installed base of Corporate Webmail Clients is expected to grow from 629 million in 2012 to over 1 billion by year-end 2016." Clearly webmail is a significant communication medium.

As more interactions become digitized, ranging from communications to finances, a number of forensic hurdles present themselves [6]. Service-oriented computing presents an interesting challenge, as we no longer see unified bodies of evidence aggregated within traditional forensic mediums. Significant research has gone into approaching specific services [7], but little work has gone into establishing a best practice approach to such evidence starting with the initial acquisition of disk-based evidence[2].

Best practices have emerged in the forensic representation of evidence. In particular, XML has become known as a medium for the creation of well-structured forms of evidence representation [8]. A well-known example of this is the Digital Forensics XML (DFXML) format, used for representing disk media as a combination of disk partitions, file systems, and file metadata in XML [9]. These formats facilitate the storage, authentication and analysis of evidence in various ways.

Improvements in the forensic analysis of email have largely followed that of big data — recent contributions to the field include statistical and machine learning techniques used to facilitate stylometric analyses, author attribution, and more into a cohesive analysis technique [10]. While these methods improve the analytic process of email forensics, there still lacks a holistic approach.

Similar to disk forensics, email also contains indexable metadata, in the form of headers, which can be useful to direct the focus of an analysis. From this metadata alone, an examiner can detect communication flows and evidence tampering, among other things. As shown by Banday in [11], the email headers are a valuable source of information in a forensic investigation involving email.

III. Methodology/Framework

We seek to help reduce the disparity between the analysis methods available for disk-based evidence and those available for web-based evidence such as email. To that end, our process for the acquisition and storage of online evidence makes available the means whereby analysis tools can handle and analyze such evidence.

In brief, our process consists of discovering online credentials from acquired evidence, mapping those credentials to their corresponding services, extracting evidence from each service, authenticating and processing that evidence into a standardized representation format, and then performing the actual analysis. Fig. 1 depicts this flow. We now discuss each part of the process.

A. Initial Acquisition

As in any investigation, once the examiner has secured the evidence, the first step is to acquire a "forensic copy," which for all purposes is an exact duplicate of the original. Forensic copies serve as a protection for the original evidence since the examiner works with these instead of the originals, allowing them to perform analyses without the risk of compromising the integrity of the evidence.

During the initial acquisition of a hard drive, the data that is available is mostly limited to the structure of the drive as found in the Master Boot Record (MBR) or in one of the Volume Boot Records (VBRs). From these structures, the examiner can also extract additional information about the file system for a particular volume, but again this only provides structural information, such as which sectors on the disk store parts of a file. At this stage, there is no indication of where any information related to the suspect's online activity and accounts may reside on the disk. For this reason, initial acquisition requires the additional steps of credential discovery, evidence mapping, and supplemental acquisition, as we will now describe in Sections III-B, III-C, and III-D, respectively.

B. Credential Discovery

In our process we use the term "credential" to denote any data which can identify the owner of the credential (e.g. the suspect) in some useful way. The breadth of this definition allows for its application without respect to the format in which the data is stored or the type of service to which it is mapped. Also, while other terms indicate a similar idea, such as "footprint" [12], "fingerprint", and "profile", none of these convey their purpose within our process, which is to reestablish a connection with online services to extract evidence. To this end we define a phase, "credential discovery", in which we detect credentials stored in a piece of evidence which an examiner can use to further recover additional evidence for the investigation.

The formats of credentials range from simplistic (text files containing user names and passwords) to complex (session cookies), and discovering all types of credentials will require

[2] By "disk-based evidence" we mean to include all forms of digital evidence that have more traditionally been part of an investigation, not just hard drives.

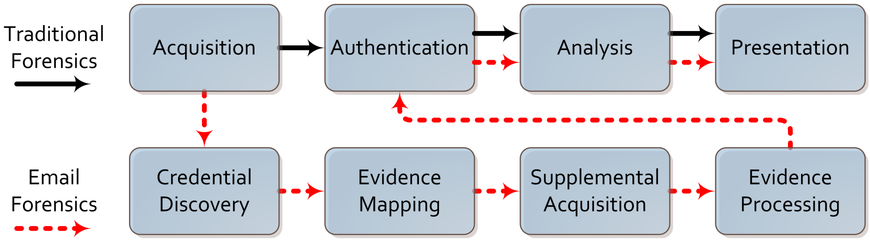


Figure 1: The traditional forensic workflow combined with the email approach

an equally diverse set of approaches. Some possible credential discovery approaches include:

- **Brute force:** Given a set of criteria (such as a regular expression) for what may possibly be credentials, linearly search the evidence, including any file slack or bad sectors. This has the disadvantage of being neither intelligent nor efficient.
- **Search known locations:** Search for files known to regularly store credentials, such as key ring databases, cookie files or databases, registry entries, etc. While more efficient, this has a much narrower scope and may overlook legitimate credentials.
- **Heuristic-based:** Using machine learning or similar techniques, learn through past experiences and feedback from the examiner what constitutes a usable credential when investigating the raw data.

Practitioners may develop other approaches, and each approach may have varying levels of success on different data sets. As such, it may be necessary to use all available approaches on each data set, depending on the computational resources available. In the best case, a large-scale, distributed system would be utilized with as many approaches as possible including proprietary internal tools, remotely hosted tools, and open source tools to discover the largest possible set of credentials.

C. Evidence Mapping

Following the discovery of credentials, it is necessary to map them to a source of evidence before performing any further acquisition. In other words, this mapping determines what online service the suspect accesses using the credential. Depending on the approach taken to discover a set of credentials, the form in which they were stored, and any accompanying data stored with the credentials, the difficulty of the mapping process may vary. Examples include email addresses or cookies which specify the domain to which they belong, spreadsheets organized in a manner which makes this information evident to an examiner, or a text file may store a user name and password with no indication of the service for which they are valid. Manual examination may be necessary to complete the mapping process if this is the case.

D. Supplemental Acquisition

After mapping a credential to a service, the next step is to acquire a forensically sound copy of the service's data. To help justify the use of certain acquisition methods during this phase, it may be useful to draw parallels between different types of traditional forensic acquisition and the circumstances characterizing supplemental acquisition from online sources. The two types of acquisition are static and live, which correspond to acquiring data from unchanging or volatile systems, respectively. When applied to performing forensics on an email account, a static acquisition is equivalent to acquiring data from stored Personal Storage Table (PST) files, frozen accounts, or logs from journaling or Simple Mail Transfer Protocol (SMTP) servers, whereas a live acquisition is equivalent to acquisition performed on active email accounts via Internet Message Access Protocol (IMAP) or using other methods, running servers (SMTP, journaling, etc.), or webmail services.

During the acquisition process, examiners must follow established best practices for any data source from which they extract evidence — this can be best achieved by defining an engine that utilizes modules which meet the requirements of *the rules of evidence* to acquire this data. To ensure the process is repeatable, we treat this portion of the process as a black-box engine which follows a set of steps to present the recovered data in a source-agnostic form so that the next module in the engine can process the evidence without regard to the source from which it originates. This engine should *reuse* the credentials previously discovered, *acquire* the most complete representation of the email (including headers and body), and then *store* them as a separate copy in an intermediate format for the purpose of evidence processing into a format which examiners will later use. Once these steps have become well defined, automation becomes trivial and should be implemented as a means to prove that the process is repeatable and forensically sound.

The nature of online storage requires careful consideration of data integrity and authentication issues; it is infeasible to represent the data exactly as stored at the remote location using our process; however, the focus of this process hinges on the text-based content of email evidence (including the headers and the body of the message) and not on the structure of the data stored on disk. As each email is a discrete, individually identifiable piece of data, we assert that a checksum of the plain text content of the original form of an email message is the most useful check against data integrity. As the acquisition process is automated, repeatable, and the data yielded is verifiable using a hash, we present the evidence acquired in this phase as a forensic copy of the evidence in an intermediate format.

E. Evidence Processing and Authentication

To increase the value of the intermediate representation mentioned in the previous step, it is necessary to facilitate a common evidence representation for the acquired data. To simplify working with this data, the representation should retain metadata about the data source as well as the data itself, so as to clearly identify any specific characteristics of the data that would be important during the reporting process. Using a common representation also adds the benefit of being able to develop tools which treat the evidence in a source-agnostic manner, since the representation abstracts away the differences between webmail and locally stored emails, simplifying the development and validation/verification process of forensic tools.

A well-structured format lends itself to the above goals, allowing for easy searching, classification, and general use of the evidence while providing an extra layer of abstraction from the raw evidence to help maintain the forensic integrity of the information. When using a structured representation, an examiner can employ verification techniques (such as schema verification) to prove the accurate representation of the evidence. Such a format and verification techniques also lends to the use of our process in a collaborative environment; for example, an organization may provide its implementation of an analysis tool remotely through a Service Oriented Architecture (SOA) implementation or multiple organizations may create separate, yet interoperable, tools using a common evidence representation.

In order to properly authenticate the data after acquisition and processing, the evidence representation format used should store the checksum generated during the acquisition phase. By storing this checksum, examiners will be able to confirm that the integrity of the data has not changed since it was first acquired, or if necessary and possible, they can perform a subsequent acquisition against the online service to check for changes to the available data or verify the accuracy of the first acquisition attempt.

F. Analysis

The next step after acquiring, processing, and authenticating the evidence is to perform forensic analyses that will be informative for the purposes of the investigation. Since our methodology only provides a process for the acquisition and storage of supplemental evidence, the implementation of new analysis tools is beyond the scope of this work. However, our methodology reduces the amount of effort required for analysis of online evidence in two ways. First, our methodology removes the need to manually acquire supplemental evidence as part of the examination workflow. The steps of discovering credentials, mapping them to online services, acquiring data from the service, and processing the data into a standard format are all performed automatically, saving time while providing increased breadth to the incident report.

Second, our methodology specifies that the standardized data storage format should have a way by which to validate its structure. Three benefits arise from this requirement: 1) acquired data in a validated format gives tool developers confidence in the structure and type of the data; 2) developers do not need to write analysis tools to handle multiple formats, since it is possible to convert evidence to the format used in the process, making tools more reusable; 3) a comparison of the output from multiple tools allows for checking accuracy[3] or for evaluating performance.

With these benefits, our approach provides significant advantages in collaboratively discovering, collecting, and analyzing evidence stored by online services.

IV. Implementation Details

To demonstrate our framework, we now give the details of our plugin-based forensics framework for online evidence, called PlugsE[4].

PlugsE is a framework implemented in Python meant to act as the black box engine mentioned in Section III-D consisting of separate modules to handle each step of the forensic process and a backbone which integrates them into a seamless tool. It has been developed with extensibility in mind, where adding a specific implementation of any step in our process is achieved by a system administrator manually adding entries to one of four manifest files which specify to the PlugsE backbone the name of the module, the type of data (DFXML file, Google cookies, keyring file, etc.) it handles, as well as how to access the module from a programmatic standpoint. The access vector could be, for example, a command-line executable or a service available via a Remote Procedure Call (RPC) interface such as REST. PlugsE stores a manifest file for each step in the forensic process and parses them to create a vector table which the backbone uses to map the different types of data it is presented with to a specific implementation of a step. Through the use of these manifests, each step in the forensic process can be viewed as a collection of modules which implement differing approaches to the specific forensic task at hand.

Each module must both accept as input and return as output JavaScript Object Notation (JSON) representations of the data being acted upon coupled with logging information (start/end times, checksums, module name and version), which aids in providing a common representation of data within the system, facilitating interoperability of modules written by different developers, organizations, or even in distinct languages.

This modular approach offers a number of interesting benefits including that a developer can implement a number of data flows within our forensic process and a step in the forensic process may be offloaded to a remote server via RPC to a module provided by another organization in a SOA fashion, with the backbone and examiner being oblivious to the geographical location or implementation details of the web service. These qualities may benefit a distributed, collaborative

[3] As discussed in [13], validating forensic tools by comparing their output is important, but requires executing the tools against the same evidence.

[4] Available at https://bitbucket.org/jpaglier/plugse

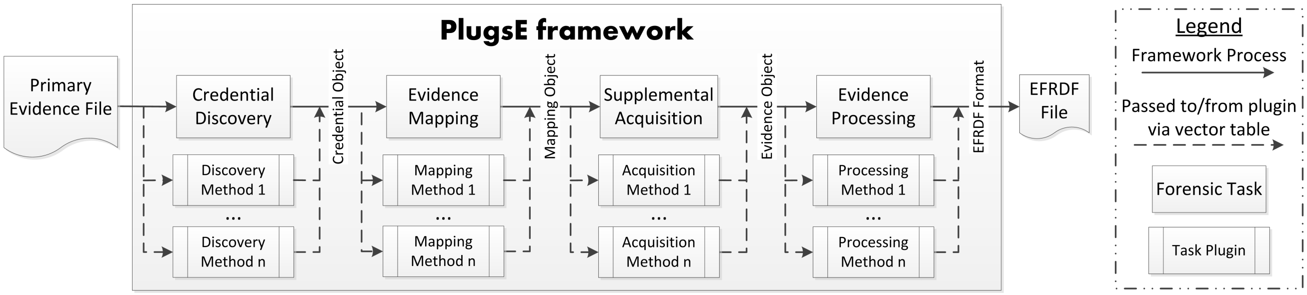


Figure 2: The PlugsE framework

approach to forensics, such as the one laid out in the CUFF framework [14].

As a proof of concept, we now show how to use PlugsE and our online evidence acquisition steps from Section III to retrieve the contents of a Gmail account using cookies containing session information that is still valid.

A. Initial Acquisition

In our implementation, we make the assumption that an examiner has already completed the work of initial acquisition (as described in Section III-A) of a hard drive from a desktop computer and created a forensic copy. Ideally, this would be performed using a system such as the one presented in [14], which allows for the analysis of evidence to automatically begin after acquisition. Also, the modules in our implementation depend on the DFXML representation of the evidence, so the examiner (or the tools used by the examiner) must ensure its creation in this phase.

B. Credential Discovery

With a forensic copy of the target device accessible, it is now possible to begin searching for credentials. By creating a PlugsE discovery module, Henson[5], that searches for browser cookies utilizing a **Search known locations** approach, we easily discover the cookies for the Chrome browser on a Windows machine at `%USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\Cookies`. While other browsers' cookies are also in known locations, this file is the focus of our proof of concept.

We adopt a straightforward approach to searching for the existence of a known path. It takes as input a list of paths for which to search. First, it decodes all of the paths, meaning it resolves any Windows system variables to all matching explicit paths. Then it splits each path into its subdirectories and iterates through them, checking for their presence in a representation of the filesystem's structure created beforehand from the DFXML file. If the full path exists, this is recorded for later use.

The complexity of our algorithm is $O(n \cdot m)$, where $n = |resolved\_paths|$ and $m = |dir\_contents|$[6]. We make the assumption that the number of subdirectories in a given path will be limited and add no more than a constant multiplier to our algorithm's complexity because we are searching for known paths of common programs, meaning it does not have the same capacity for expansion the way that n or m do. For example, in the case of Chrome cookie files in Windows 7, the path specified previously will resolve to `C:\User\<user name>\AppData\Local\Google\Chrome\User Data\Default\Cookies`, which is a total of 9 directories before reaching the target file (`Cookies`).

```
[
 {
  "source":"dfxml://file37",
  "format":"cookie",
  "md5"   :"1e6c344157eb14a79fefc07a9800695c",
  "found" :"2013-02-28T16:55:42-07:00"
 }
]
```

Figure 3: Initial mapping structure created by a discovery module

After discovering the cookie database, the last task the module performs is to store important information about the possible credential source in a JSON file for use in the Evidence Mapping phase. While the discovery module cannot map the credential to a service because it did not search the contents of the database for service-specific information, it stores the source, format, checksum, and time of discovery of the credential in a JSON file as illustrated in Fig. 3. With this, PlugsE's logging process has the information it needs and the relevant evidence mapping modules know which files to use when carrying out their discovery attempts. The module then returns the JSON file to the main PlugsE process which passes it to any modules registered for handling a cookie credential.

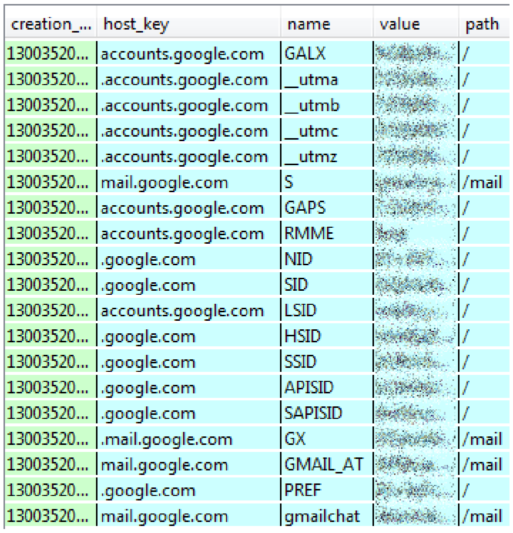
C. Evidence Mapping

Now that the cookies have been discovered, PlugsE invokes all evidence mapping modules registered to work with cookie databases, passing the possible credential sources to each of them. In some cases, it may be necessary at this point for the examiner to manually map the credentials to a service, as mentioned in Section III-C. PlugsE will determine that this is the case when one of two things happens: 1) no mapping module has been registered to work with the source and format of a credential, or 2) none of the registered modules were successful in mapping it to a service.

In our example, identifying the cookies for a Gmail account is straightforward because the fields shown in Fig. 4 will

[5] Available at https://bitbucket.org/jpaglier/henson

[6] Due to the page limit, we omit the details of our algorithm.



creation_...	host_key	name	value	path
13003520...	accounts.google.com	GALX	[redacted]	/
13003520...	.accounts.google.com	__utma	[redacted]	/
13003520...	.accounts.google.com	__utmb	[redacted]	/
13003520...	.accounts.google.com	__utmc	[redacted]	/
13003520...	.accounts.google.com	__utmz	[redacted]	/
13003520...	mail.google.com	S	[redacted]	/mail
13003520...	accounts.google.com	GAPS	[redacted]	/
13003520...	accounts.google.com	RMME	[redacted]	/
13003520...	.google.com	NID	[redacted]	/
13003520...	.google.com	SID	[redacted]	/
13003520...	accounts.google.com	LSID	[redacted]	/
13003520...	.google.com	HSID	[redacted]	/
13003520...	.google.com	SSID	[redacted]	/
13003520...	.google.com	APISID	[redacted]	/
13003520...	.google.com	SAPISID	[redacted]	/
13003520...	.mail.google.com	GX	[redacted]	/mail
13003520...	mail.google.com	GMAIL_AT	[redacted]	/mail
13003520...	.google.com	PREF	[redacted]	/
13003520...	mail.google.com	gmailchat	[redacted]	/mail

Figure 4: All the cookies created by logging in to Gmail

```
[
 {
  "domain":"mail.google.com",
  "type"  :"cookie",
  "format":"sqlite",
  "data"  :"dfxml://file37"
 },
 {
  "domain":"www.dropbox.com",
  "type"  :"plain",
  "format":"user_pass",
  "data"  :["guy@email.com",
                  "12345"]
 }
]
```

Figure 5: Mapping example with two entries.

be present. Our mapping module for PlugsE searches for these fields and upon detecting them creates an entry in the mapping table which identifies this cookie database as containing credentials for Gmail. Fig. 5 shows what this entry looks like. Although the complexity of our module depends on the efficiency of the Python sqlite3[7] library, it only searches for a constant number of keys in the target and stops searching when any key is not present, which means it contributes no more than $O(1)$ complexity to the library's operations.

D. *Supplemental Acquisition*

As we mentioned in Section III-D, tool developers must determine the best practice for acquiring data stored by a distinct online service based on the type of credential(s) discovered previously and the service's features. For our proof of concept with Gmail, we researched what features are available when we only have the browser cookies to log in. While the optimal acquisition method for retrieving a copy of all emails is to do so via IMAP, cookies are specific to the HTTP protocol and will not work to authenticate through IMAP. If plain text credentials (user name and password) were discovered, acquisition via IMAP would be possible.

Since we only have the browser cookies to work with in our proof of concept, we have a limited ability to change any account settings that will help the process of acquisition. Fortunately, Gmail allows users to grant other Gmail addresses access to their account without reentering their password. The account that is granted access is called a "delegate" and can read all the emails in the granter's account as well as send emails on their behalf. While Gmail does not provide IMAP access to grantee accounts, creating the delegate prolongs access to the target account past the two week expiration date of the cookies, allowing for any needed follow-up acquisition.

With this understanding, we wrote a pair of tools in Python to complete the supplemental acquisition. The first tool, which we call Crumbler[8], imports the cookie database from its native SQLite format to a custom subclass of the common `Cookie` Python object. The second tool automates the process of adding a delegate to an account using the Selenium[9] web driver. It opens a browser and connects to Gmail, and as long as the cookies are still valid it performs each of the steps for adding a delegate as outlined in the Google help pages[10], which takes $O(1)$ time. Once this process has completed, the grantee can access the target account by logging in to Gmail, clicking on their email address in the top right hand corner of the screen, and selecting the target account from the list of accounts to which they have access.

The final challenge to acquiring data from Gmail is that the only method for retrieving the raw email data is to essentially "screen scrape" the pages returned during a web session, parsing through the HTML and using regular expression patterns or searching through the Document Object Model (DOM) for the desired elements. A tool is currently under development for the purpose of downloading the contents of a Gmail account. The messages should then be processed into a standard format, as we discuss in the following section. We recognize that a few circumstances have to be ideal in order for this acquisition process to work, namely that the owner of the credentials is always signed in, that the cookies have not yet expired and are discoverable by some means, and that the notification banner of having added the delegate account will not compromise the investigation. It is inevitable to retain these circumstantial dependencies. However, we also assert that those investigations for which they do not hold have not lost access to evidence that otherwise would have been accessible, while those for which they do hold have gained access to a great source of information[11].

E. *Evidence Processing and Authentication*

To demonstrate our evidence processing phase, we must convert our intermediate representations to a well-structured

[7] http://docs.python.org/2/library/sqlite3.html; complexity of individual operations not provided.

[8] Available at https://bitbucket.org/mmabey/crumbler

[9] http://seleniumhq.org/

[10] http://support.google.com/mail/answer/138350

[11] Either way, to achieve comprehensive forensic analysis on email evidence, we believe such an approach is necessary and beneficial.

```
<?xml version=’1.0’ encoding=’ISO-8859-1’?>
<mailbox type="mbox">
        <message>
                <Subject>&lt;![CDATA[[dovecot] Re: some problem with dovecot]]&gt;</Subject>
                <Date>
                        <year>2003</year>
                        ...
                </Date>
                <From>
                  <sender>
                        <alias>&lt;![CDATA[Jesse Peterson]]&gt;</alias>
                        <email>jpeterson275@attbi.com</email>
                  </sender>
                </From>
                <non-standard>
                  <X-Original-To>dovecot@procontrol.fi</X-Original-To>
                </non-standard>
                <byte_runs file_offset="1101673" len="2159"/>
                <checksum>
                  <md5>8e4bb7462b991183cf5b2adc87970227</md5>
                </checksum>
        </message>
</mailbox>
```

Figure 6: An Abbreviated Sample of the EFXML Schema

format which follows the current best practices of the forensic process. During the development of our proof of concept, we surveyed the strengths and weaknesses of existing formats and concluded that the mbox format [15] was the best suited to our purposes. The mbox format is a flat-file, plain text representation of email which is easy to parse and human readable; these traits greatly reduce the time needed for examination of evidence and development of tools, both of which are costly in terms of resources and time. Furthermore, normally mbox stores attachments in some form of directory structure related to their messages so that attachment analysis could be started as part of an automated process, separate from the email data. Finally, using mbox is useful even when processing the common PST format as tools from libPST[12] provide the conversion. Finally, we assert that mbox is valid for use as a forensic copy format as it is “output readable by sight, shown to reflect the data accurately” and thus “is an original” [2, pg. 162], so long as the acquisition process used was sound.

A current trend in digital forensics is the use of XML as a data representation format, allowing for a firm layer of abstraction “between feature extraction and analysis” and “a single, XML-based output format for forensic analysis tools” [8]. For evidence representation in existing methodologies, DFXML is a standard to represent disk, partition, file system, and file data in a unified manner [9]. A major benefit of DFXML is the generality of its representation; regardless of disk geometry or forensic copy format, the evidence is represented in the same manner.

When analyzing email evidence, the most significant metadata is contained within the header of the email itself. Email headers include information such as the sender and recipient (From and To), unique message identifiers (Message-ID), reply addresses (Reply-To), and more [16]. Even without considering the content or body of emails, these headers have been shown to be useful in forensic investigations as a means to achieve author attribution, detect attempts to obfuscate sequences of events during a time period of interest [11], as well as identify communication flows and perform social networking analysis.

Although DFXML is quite efficient for representing massive amount of data from a filesystem, it is ill-suited for storing the header information of emails, as file system metadata is not closely related to the analysis of evidence contained within email messages. While it could be extended to fit this purpose, the resulting format would become overly encumbered and its size efficiency greatly reduced. Because of this, we have defined two new representations which are more suitable for email forensics, but maintain some of the standard elements introduced in DFXML, such as byte runs of discrete pieces of evidence. We call these new formats Email Forensics XML (EFXML) and Email Forensics Resource Description Framework (EFRDF). While these two representations are functionally equivalent, we designed them with different purposes in mind. While EFXML lends itself mostly to syntactic reasoning methods, EFRDF is meant to be used when examining the highly semantic nature of information contained within email evidence. More importantly, investigations utilizing multiple bodies of email evidence which, when combined, may reveal complex information flows through social interactions may benefit from semantic analysis. EFRDF is based upon the Resource Description Framework standard [17], a subset of XML often used for semantic representations of data.

As email is text-based and can be easily represented in XML without complicated encoding, EFXML and EFRDF present many of the same benefits presented by Alink et al. [8] and Garfinkel [9], including easy searching and classification of information. As an added benefit of using an XML-based

[12] http://www.five-ten-sg.com/libpst/

format, EFXML and EFRDF have clearly defined schemas which can verify the output from tools that generate these formats. As an example of their differing, yet equivalent, representations, a To field reading:

```
To: jsmith@gmail.com
```

would yield the EFXML element:

```
<To>
        <recipient>
                <email>jsmith@gmail.com</email>
        </recipient>
</To>
```

or the EFRDF element:

```
<message:to>
        <recipient:email>
                jsmith@gmail.com
        </recipient:email>
</message:to>
```

which accurately represent the "To" header field in a much more structured manner, allowing for easily focusing on or excluding messages based upon their apparent recipients. Similarly, to reflect the data extraction capabilities provided by the "byte_runs" element in DFXML, we included a simplified element which details the span of bytes within the mbox file where the email resides and can be extracted from using tools such as those designed for DFXML, the Unix command dd, or other comparable programs. While line numbers would be equally useful with regard to the mbox format, we decided to use the "byte_runs" field in each representation to follow existing standards. An abbreviated sample of an EFXML representation of a mailbox can be seen in Fig. 6.

We created two tools for parsing mbox files into corresponding EFXML and EFRDF representations, named mbox2efxml[13] and efxml2efrdf[14]. We loosely based mbox2efxml on Philip Guo's create_mbox_summary tool [18], but with comprehensive support for the email headers specified in [16] and well-formed XML through the use of the lxml library[15].

Because the mbox2efxml tool only works with mbox archives, it assumes that if the email was originally in a different format, some tool has already made the conversion to mbox. For example, libPST's readpst[16] tool reads in a PST file and outputs a separate mbox file for each folder contained within the PST. These separate files are then iterated over and each message within them is processed. Our tool runs in $O(m \cdot n \cdot q)$ time where $m = |mbox\_paths|$, $n = |messages|$ for all mailboxes, and $q = |headers|$ for all messages.

The efxml2efrdf tool works in a similar fashion to mbox2efxml by reading in an EFXML file using the lxml library and converting the contents to its EFRDF equivalent. While leaving out the body of the email may limit the scope of a semantic reasoning approach, we leave explorations into the costs and benefits of its exclusion to future work, where we will be able to further investigate the subtleties of a semantic approach. The structure of EFXML and EFRDF as specified in their schemas helps overcome one shortcoming of mbox as it relates to forensics, which is that there is no way to add metadata to the file. Of particular interest are the fields which help maintain the chain of custody by storing information on the name of the program that created the mbox, EFXML, and/or EFRDF files; the version of the programs; the date and time of their creation; the target email address; the size of the mbox file; and MD5 and SHA1 checksums for the mbox file. With this information, it is possible to keep track of how the evidence was acquired, authentication information for the entire set of emails[17], and what programs handled the evidence at what time, all of which are required by the rules of evidence.

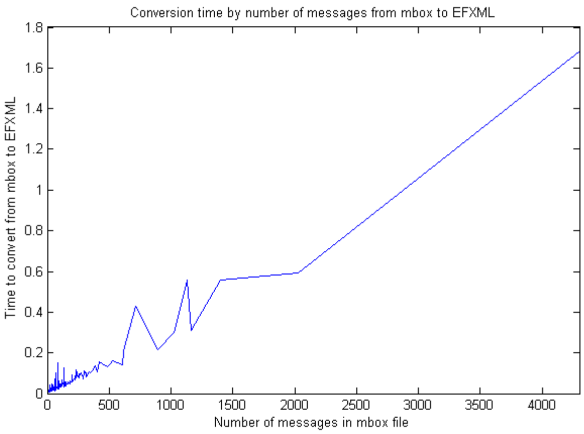


Figure 7: Conversion times of single mbox files in the Enron data set by number of messages

F. Analysis

Our approach creates well-defined, structured, and verifiable representations of email data. Since they are XML formats, developers can easily craft tools and validate them using common XML parsing libraries to facilitate the analysis process, much like DFXML. Also, with the intermediate mbox format and the EFXML/EFRDF abstractions, forensic analyses can be carried out while the forensic copy remains intact, regardless of the data source's original storage format. The development of analysis tools is beyond the scope of this paper.

V. Evaluation

As differing implementations of each step of the forensic process will vary based on data source, we focus mainly on two factors: (1) the efficiency of EFXML and (2) the running time of sample implementations of each step to show an example of our process functioning in a useful manner.

[13] Available at https://bitbucket.org/jpaglier/efxml

[14] Available at https://bitbucket.org/jpaglier/efxml2efrdf

[15] http://lxml.de/index.html

[16] http://www.five-ten-sg.com/libpst/rn01re01.html

[17] Unless the target account has been frozen by the provider, acquiring emails from the same account at two different times will likely yield two distinct data sets, preventing the checksums from matching. As such, the checksums are provided for integrity checks against the same mbox archive to ensure it does not change while being analyzed and not against past or future acquisitions.

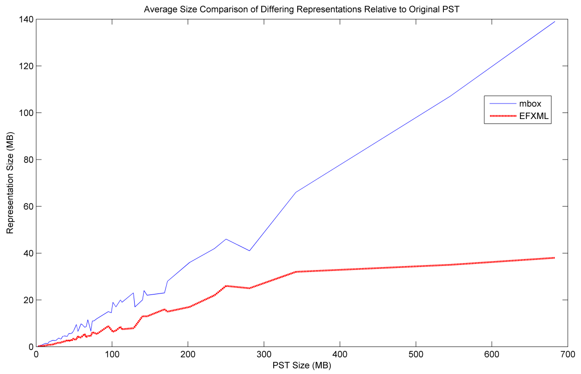


Figure 8: Average Size of mbox and EFXML Representations Relative to PST in the Enron Data (Normalized)

Initially, a test of the **Search known locations** algorithm was tested and was confirmed to be roughly linear, taking about 2 milliseconds for every 100 file objects listed in a DFXML record. When the process was carried through to evidence mapping, another 10 seconds per discovered Chrome cookie database containing Gmail cookies was added.

To evaluate the Email Forensics XML format and the evidence processing step, we designed an experiment which compares the size of Personal Storage Table (PST) files, their respective mbox files, and their EFXML representations. The PST files used were the publicly available Enron Corporation email data[18], which measures 8.70GB across 148 mailboxes, containing a total of 517,431 messages and 3,299 folders (once decompressed).This dataset does not include attachments or certain redacted portions of the original data. Each PST file was then converted into its respective mbox representation using the readpst tool available through libPST, yielding 1.2GB of text data. Then these mbox files were processed using the mbox2efxml tool discussed in Section IV-E generating 614MB of data after approximately 3.5 minutes of processing on an Acer Aspire 4830T[19].

When recording the time taken to process each mbox file individually, as seen in Figure 7, the average processing time per message was 0.57ms. Figure 7 also shows a roughly linear increase in processing time with regard to the number of messages, confirming our estimate of $O(m{\cdot}n{\cdot}q)$ running time. A comparison relative to mbox size, after conversion from PST, is depicted in Figure 8, which visually demonstrates that the file sizes are positively correlated across this dataset. There was a notable decrease in file size when converting from PST into mbox, which may be explained by the block allocation scheme used in the PST format.

Following the step of processing the evidence into EFXML, a number of verification tasks were carried out including reproducing checksums and comparing counts of messages between the original and EFXML representations. Due to the nature of email data, it is possible to observe a size increase in particularly imperfect cases (e.g. where volume of header data exceeds the volume of body data) after the addition of the EFXML tags to the data[20]; however, our evaluations point toward an average case which does not approach this situation. Potential size decreases could be seen when considering attachments in the body of evidence.

VI. Discussion

We now discuss our work in relation to the two important topics of the rules of evidence and collaboration.

A. *Rules of Evidence*

Any forensic framework must uphold *the rules of evidence* (authenticity, admissibility, completeness, and accuracy/reliability), which are the canonical guidelines for handling evidence. The following is a discussion of how our approach supports these important principles.

The authenticity of evidence from online sources relies on the same arguments as other live acquisition methods, which by nature are difficult to verify [19]. While our framework is capable of facilitating the protection of evidence authenticity, it remains an issue for developers to implement sound acquisition plugins for PlugsE that capture an accurate representation of the acquired data.

Admissibility implies two key concepts: 1) that the evidence was acquired following proper procedures, and 2) that it was handled properly after acquisition. As we discussed in Section I-A, it is the responsibility of practitioners to understand their duties with regard to the first item. However, our framework facilitates the second item by using the EFXML and EFRDF files to record details about the tools that acquired and processed the evidence, which examiners can include in chain of custody forms when necessary.

For evidence to be complete, the format into which it is acquired needs to accurately reflect the original data, as we mentioned in Section IV-E. Without cooperation from the service provider, it is impossible to know the exact structure of the original data, so we must instead focus on retaining the available data using standards such as RFC 4021 [16] as a guide. Even though our tools may omit newlines, control characters, and other non-essential data during the processes of acquisition and conversion to mbox, we preserve all the header information, body text, and attachments, which are the portions of interest and which will hold sway in a legal setting. Furthermore, we reiterate that our approach provides examiners access to relevant evidence not available after a simple acquisition of a hard drive, aiding the examiner to form a more clear, complete, and well-informed report on the suspect. We do, however, feel that the algorithm discussed in Section IV-E is costly and should be improved upon in future work. This may be achieved through defining and extracting only specific headers which reveal important information flows (reducing a linear factor to a constant), using semantic reasoning to

[18] Acquired from http://www.enrondata.org

[19] http://www.cnet.com/laptops/acer-aspire-timelinex-4830t/4505-3121_7-35029979.html

[20] We did, in fact, encounter such a case using a sample acquired at http://www.dovecot.org/tmp/dovecot-crlf

determine which messages to focus on, or other yet unexplored tactics. As EFXML and EFRDF are structured data formats, they may facilitate these future approaches by simplifying the development process of tools which utilize them and following current trends in the forensics community [8].

Our implementation ensures the reliability and accuracy of evidence it handles by measuring the integrity of each message by taking its checksum during supplemental acquisition and evidence processing. With this, an examiner may verify the evidence has remained unchanged after each step of the process. Also, because we provide schemas for EFXML and EFRDF, developers can ensure the reliability of their own tools more easily.

B. Collaboration

Our approach facilitates collaboration in many ways. First, it presents a platform for tool developers that allows them to focus on building their algorithms and modules correctly without having to devote precious time to the acquisition of email data or handling its native format, much the same way that DFXML has done for disk forensics [20]. Second, the use of language agnostic formats such as JSON, EFXML, and EFRDF allows for interoperability between modules in the system, allowing for both collaborative development and the sharing of work among developers with different technical backgrounds as well as allowing for organizations to provide their implementations as SOA products without revealing the intimate details of their methodologies, which may reveal trade secrets or other proprietary information. The ability to share work and tools allows for scenarios where different examiners within one or more organizations can take responsibility for different steps in the forensic process, which could be further enhanced by use in a collaborative forensic system such as CUFF [14]. Finally, the use of PlugsE manifest files gives the system a degree of autonomy, where an administrator need not have detailed programming expertise, and further increases interoperability by abstracting the details of how modules are accessed; to a practitioner the use of locally-hosted command line tools becomes no different from using a remotely-hosted SOA product accessed over an RPC protocol while logging information contained within their outputs maintains the chain of custody.

VII. Conclusion

In this paper, we have defined a general methodology for carrying out email forensics and shown a proof of concept implementation with evaluation results. In our approach, we broadened the definition of credentials, identified methods for discovering credentials, and demonstrated the need for a generic evidence representation. Our implementation has shown an example of credential discovery for Gmail accounts, a method for reestablishing existing Gmail sessions, the steps needed to carry out a supplemental acquisition, and a completed evidence processing phase generating both an intermediate mbox representation of email evidence and proposed EFXML/EFRDF representations of email headers upon which further analysis can be carried out while addressing the need to facilitate collaboration amongst developers and organizations during the creation of tools for forensic investigations as well as the investigations themselves.

References

[1] Fox News, "Petraeus resigns after affair with biographer turned up in fbi probe, fox news confirms," http://www.foxnews.com/, November 2012.

[2] B. Nelson, A. Phillips, F. Enfinger, and C. Steuart, *Guide to computer forensics and investigations*. Boston, Mass: Thomson Course Technology, 2008.

[3] K. Ashton, "That 'Internet of Things' Thing," *RFiD Journal*, vol. 22, pp. 97–114, July 2009.

[4] R. Littlehale. (2013, March) Hearing on ECPA part 1: Lawful access to stored content, written testimony of Richard Littlehale. http://judiciary.house.gov/hearings/113th/03192013_2/Littlehale%2003192013.pdf. Tennessee Bureau of Investigation.

[5] Radicati Group, Inc., "Email Market, 2012-2016," http://www.radicati.com/wp/wp-content/uploads/2012/10/Email-Market-2012-2016-Executive-Summary.pdf, October 2012.

[6] S. Greengard, "On the digital trail," *Communications of the ACM*, vol. 55, no. 11, pp. 19–21, November 2012. [Online]. Available: http://doi.acm.org/10.1145/2366316.2366323

[7] D. Quick and K.-K. R. Choo, "Dropbox analysis: Data remnants on user machines," *Digital Investigation*, vol. 10, no. 1, pp. 3 – 18, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S174228761300011X

[8] W. Alink, R. Bhoedjang, P. Boncz, and A. de Vries, "XIRAF–XML-based indexing and querying for digital forensics," *Digital Investigation*, vol. 3, pp. S50–S58, 2006.

[9] S. Garfinkel, "Digital forensics XML and the DFXML toolset," *Digital Investigation*, vol. 8, no. 3–4, pp. 161–174, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287611000910

[10] R. Hadjidj and et al., "Towards an integrated e-mail forensic analysis framework," *Digital Investigation*, vol. 5, no. 3–4, pp. 124–137, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287609000036

[11] M. T. Banday, "Analyzing e-mail headers for forensic investigation," *Journal of Digital Forensics, Security, and Law*, vol. 6, pp. 49–64, 2011. [Online]. Available: http://www.jdfsl.org/subscriptions/abstracts/JDFSL-V6N2-column-Banday.pdf

[12] S. Garfinkel and D. Cox, "Finding and archiving the internet footprint," in *Digital Lives Research Conference: Personal Digital Archives for the 21st Century*, February 2009.

[13] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digital Investigation*, vol. 6, no. Supplement 1, pp. S2–S11, 2009, the Proceedings of the Ninth Annual DFRWS Conference. [Online]. Available: http://www.sciencedirect.com/science/article/B7CW4-4X1HY5C-3/2/090ebc16025d598c775d87c8abbb7ae5

[14] M. Mabey and G.-J. Ahn, "Towards collaborative forensics: Preliminary framework," in *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, 2011.

[15] E. Hall, "The application/mbox Media Type," RFC 4155 (Informational), Internet Engineering Task Force, Sep. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4155.txt

[16] G. Klyne and J. Palme, "Registration of Mail and MIME Header Fields," RFC 4021 (Proposed Standard), Internet Engineering Task Force, March 2005, updated by RFC 5322. [Online]. Available: http://www.ietf.org/rfc/rfc4021.txt

[17] W3C, "RDF primer," W3C Recommendation 10 February 2004, W3C, February 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[18] P. Guo, "Email analysis scripts for mbox mailbox files," http://www.pgbovine.net/mbox-analysis.htm, Sep. 2006.

[19] B. Schatz, "BodySnatcher: Towards reliable volatile memory acquisition by software," *Digital Investigation*, vol. 4, Supplement, no. 0, pp. 126–134, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287607000497

[20] S. Garfinkel, "Automating disk forensic processing with sleuthkit, xml and python," in *Systematic Approaches to Digital Forensic Engineering, 2009. SADFE '09. Fourth International IEEE Workshop on*, 2009, pp. 73–84.